Java Assignment

Q1. Explain the concept of Object-Oriented Programming (OOP) and its benefits. What is inheritance? Provide a code example in Java.

Ans. Object-Oriented Programming (OOP) is a programming style centered around "objects." Objects represent entities, which may be a real world user or a concept, with attributes (properties) and behaviors (methods). The main principles of OOP are:

1. **Encapsulation**: Bundling data (attributes) and methods (functions) that operate on the data into a single unit or class.
2. **Abstraction**: Hiding the complex implementation details and showing only the necessary features of an object.
3. **Inheritance**: Creating a new class from an existing class to reuse, extend, or modify the behavior of the parent class.
4. **Polymorphism**: The ability of different objects to respond, each in its own way, to identical messages (methods).

Benefits of OOP

1. **Modularity**: Code is organized into discrete objects, making it easier to manage.
2. **Reusability**: Existing objects can be reused across different programs.
3. **Scalability**: Easier to scale programs by adding new objects.
4. **Maintainability**: Easier to update and maintain code.

Inheritance

Inheritance allows a new class (subclass) to inherit properties and methods from an existing class (superclass). This promotes code reuse and establishes a natural hierarchy. Example:

```
class Animal {
    // Attributes
    String name;
    int age;
    Animal(String name, int age) {
        this.name = name;
        this.age = age;
    }
    void makeSound() {
        System.out.println("Some sound...");
    }
}
class Dog extends Animal {
    Dog(String name, int age) {
        super(name, age);
    }
    @Override
    void makeSound() {
        System.out.println("Bark");
    }
}
public class Main {
    public static void main(String[] args) {
        Dog myDog = new Dog("Buddy", 3);
        myDog.makeSound();
        System.out.println("Name: " + myDog.name + ", Age: " + myDog.age);
    }
}
```

Q2. Write a Java program to calculate the factorial of a number.

Algorithm

1) Get Number from the user
2) loop from n to said 1 and multiply them to the result again and again
3) Print the output

Program

```
import java.util.Scanner;



public class factorial {

	public static void main(String[] args) {

		Scanner scanner = new Scanner(System.in);

		int n = 0;

		System.out.println();

		System.out.print("Enter your number: ");

		n = scanner.nextInt();

		scanner.nextLine();

		int answer = 1;

		while (n > 1) {

			answer *= n;

			n -= 1;

		}

		System.out.println("The factorial is: " + answer);

		System.out.println();

		scanner.close();

	}

}
```

Sample I/O

```
C:\Users\niran\Downloads\Uni_Work\java\35_Niranjhan>java ./factorial.java

Enter your number: 10
The factorial is: 3628800
```

Q.3) Write a Java program to implement a simple calculator.

Algorithm

1) Define functions which take numbers, perform arithmetic operations and return them.
2) Get 2 numbers and operation from the user in the main function
3) Use switch case to call functions appropriately
4) Print the Output

Program

```
import java.util.Scanner;

public class Calculator {

    public static double add(double a, double b) {
        return a + b;
    }

    public static double subtract(double a, double b) {
        return a - b;
    }

    public static double multiply(double a, double b) {
        return a * b;
```

```
}

public static double divide(double a, double b) {
    if (b == 0) {
        System.out.println("Error! Division by zero is not allowed.");
        return Double.NaN;
    }
    return a / b;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    double num1, num2;
    char operator;
    double result = 0.0;

    System.out.print("Enter first number: ");
    num1 = scanner.nextDouble();
    scanner.nextLine();

    System.out.print("Enter an operator (+, -, *, /): ");
    operator = scanner.next().charAt(0);

    System.out.print("Enter second number: ");
    num2 = scanner.nextDouble();
    scanner.nextLine();

    switch (operator) {
        case '+':
            result = add(num1, num2);
            break;
        case '-':
```

```
                result = subtract(num1, num2);
                break;
            case '*':
                result = multiply(num1, num2);
                break;
            case '/':
                result = divide(num1, num2);
                break;
            default:
                System.out.println("Error! Invalid operator.");
                scanner.close();
                return;
        }

        System.out.println("The result is: " + result);

        scanner.close();
    }
}
```

Sample I/O

```
C:\Users\niran\Downloads\Uni_Work\java\35_Niranjhan>java ./Calculator.java
Enter first number: 10
Enter an operator (+, -, *, /): *
Enter second number: 20
The result is: 200.0
```

Q4) Write a Java program to perform the multiplication of two matrices

Algorithm

1) Define main function to get rows and columns of matrices
2) Check if its valid for multiplication and then get the values
3) Get values for each element of the matrices from the user
4) Loop through the first matrix and perform multiplication with second.
5) Print result to User

Program

```
import java.util.Scanner;

public class MatrixMultiplication {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of rows in the first matrix: ");
        int rows1 = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter the number of columns in the first matrix: ");
        int cols1 = scanner.nextInt();
        scanner.nextLine();

        System.out.print("Enter the number of rows in the second matrix: ");
```

```
        int rows2 = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter the number of columns in the second matrix: ");
        int cols2 = scanner.nextInt();
        scanner.nextLine();

        if (cols1 != rows2) {
            System.out.println("Matrix multiplication is not possible. Number of columns in the first matrix must be equal to the number of rows in the second matrix.");
            scanner.close();
            return;
        }

        int[][] matrix1 = new int[rows1][cols1];
        System.out.println("Enter the elements of the first matrix:");
        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols1; j++) {
                matrix1[i][j] = scanner.nextInt();
                scanner.nextLine();
            }
        }

        int[][] matrix2 = new int[rows2][cols2];
        System.out.println("Enter the elements of the second matrix:");
        for (int i = 0; i < rows2; i++) {
            for (int j = 0; j < cols2; j++) {
                matrix2[i][j] = scanner.nextInt();
```

```
                scanner.nextLine();
            }
        }

        int[][] result = new int[rows1][cols2];

        // Multiply the matrices
        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                for (int k = 0; k < cols1; k++) {
                    result[i][j] += matrix1[i][k] * matrix2[k][j];
                }
            }
        }

        System.out.println("The resultant matrix after multiplication
is:");
        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols2; j++) {
                System.out.print(result[i][j] + " ");
            }
            System.out.println();
        }

        scanner.close();
    }
}
```

Sample I/O

```
C:\Users\niran\Downloads\Uni_Work\java\35_Niranjhan>java ./MatrixMultiplication.java
Enter the number of rows in the first matrix: 2
Enter the number of columns in the first matrix: 2
Enter the number of rows in the second matrix: 2
Enter the number of columns in the second matrix: 1
Enter the elements of the first matrix:
1
2
3
4
Enter the elements of the second matrix:
1
2
The resultant matrix after multiplication is:
5
11
```

Q.5) Write a Java method to compute the determinant of an $N \times N$ matrix using recursion

Algorithm

1) Check if matrix is 1x1 or 2x2 and return the determinant directly.
2) Set determinant value to 0.
3) Iterate through first row, calculate submatrix determinants recursively using Laplace expansion.
4) Alternate signs and add terms to the determinant.

Program

```
import java.util.Scanner;


public class DeterminantRecursive {


        public static double[][] getSubMatrix(double[][] matrix, int excludingRow, int
excludingCol) {
                int n = matrix.length;
                double[][] subMatrix = new double[n - 1][n - 1];
                int r = -1;
```

```
        for (int i = 0; i < n; i++) {
                if (i == excludingRow) continue;
                r++;
                int c = -1;
                for (int j = 0; j < n; j++) {
                        if (j == excludingCol) continue;
                        subMatrix[r][++c] = matrix[i][j];
                }
        }

        return subMatrix;
}

public static double determinant(double[][] matrix) {
        int n = matrix.length;

        if (n == 1) {
                return matrix[0][0];
        }

        if (n == 2) {
                return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
        }

        double det = 0.0;

        for (int j = 0; j < n; j++) {
                double[][] subMatrix = getSubMatrix(matrix, 0, j);
```

```
                det += Math.pow(-1, j) * matrix[0][j] * determinant(subMatrix);
            }
            return det;
    }

    public static void main(String[] args) {

            Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of rows in the first matrix: ");
int rows1 = scanner.nextInt();
scanner.nextLine();
System.out.print("Enter the number of columns in the first matrix: ");
int cols1 = scanner.nextInt();
scanner.nextLine();

double[][] matrix = new double[rows1][cols1];
System.out.println("Enter the elements of the first matrix:");
for (int i = 0; i < rows1; i++) {
    for (int j = 0; j < cols1; j++) {
        matrix[i][j] = scanner.nextInt();
        scanner.nextLine();
    }
}

System.out.println("Determinant: " + determinant(matrix));
scanner.close();
```

```
        }
}
```

Sample I/O

```
C:\Users\niran\Downloads\Uni_Work\java\35_Niranjhan>java ./DeterminantRecursive.java
Enter the number of rows in the first matrix: 3
Enter the number of columns in the first matrix: 3
Enter the elements of the first matrix:
1
2
3
4
5
6
7
8
9
Determinant: 0.0
```

Q.6) Write a Java program to solve a system of linear equations using Cramer's rule

Algorithm

1) Compute the determinant of the coefficient matrix.
2) If determinant is zero, the system has no unique solution.
3) For each variable, replace the respective column in the coefficient matrix with the constants vector.
4) Calculate determinants of modified matrices and divide by the determinant of the coefficient matrix to find each variable.

Program

```
import java.util.Scanner;


public class CrammerRule {


    public static double determinant(double[][] matrix) {
        int n = matrix.length;


        if (n == 1) {
            return matrix[0][0];
```

```
        }

        if (n == 2) {
            return matrix[0][0] * matrix[1][1] - matrix[0][1] *
matrix[1][0];
        }

        double det = 0.0;

        for (int j = 0; j < n; j++) {
            double[][] subMatrix = getSubMatrix(matrix, 0, j);
            det += Math.pow(-1, j) * matrix[0][j] *
determinant(subMatrix);
        }

        return det;
    }

    private static double[][] getSubMatrix(double[][] matrix, int
excludingRow, int excludingCol) {
        int n = matrix.length;
        double[][] subMatrix = new double[n - 1][n - 1];
        int r = -1;

        for (int i = 0; i < n; i++) {
            if (i == excludingRow) continue;
            r++;
            int c = -1;
            for (int j = 0; j < n; j++) {
```

```
                if (j == excludingCol) continue;
                subMatrix[r][++c] = matrix[i][j];
            }
        }

        return subMatrix;
    }

    public static double[] solveUsingCramersRule(double[][] coefficients, double[] constants) {
        int n = coefficients.length;
        double detA = determinant(coefficients);

        if (detA == 0) {
            throw new ArithmeticException("The system has no unique solution");
        }

        double[] solutions = new double[n];

        for (int i = 0; i < n; i++) {
            double[][] modifiedMatrix = new double[n][n];
            for (int j = 0; j < n; j++) {
                for (int k = 0; k < n; k++) {
                    if (k == i) {
                        modifiedMatrix[j][k] = constants[j];
                    } else {
                        modifiedMatrix[j][k] = coefficients[j][k];
                    }
```

```
                }
            }
            solutions[i] = determinant(modifiedMatrix) / detA;
        }

        return solutions;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of variables: ");
        int n = scanner.nextInt();
        scanner.nextLine();

        double[][] coefficients = new double[n][n];
        double[] constants = new double[n];

        System.out.println("Enter the coefficients of the varaibles in order (put 0s as well): ");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                coefficients[i][j] = scanner.nextDouble();
                  scanner.nextLine();
            }
        }

        System.out.println("Enter the constants of the system: " );
```

```
            for (int i = 0; i < n; i++) {
                constants[i] = scanner.nextDouble();
                scanner.nextLine();
            }

            try {
                double[] solutions = solveUsingCramersRule(coefficients,
constants);

                System.out.println("The solutions are:");
                for (int i = 0; i < n; i++) {
                    System.out.println("x" + (i + 1) + " = " + solutions[i]);
                }
            } catch (ArithmeticException e) {
                System.out.println(e.getMessage());
            }

            scanner.close();
        }
}
```

Sample I/O

```
C:\Users\niran\Downloads\Uni_Work\java\35_Niranjhan>java ./CrammerRule.java
Enter the number of variables: 2
Enter the coefficients of the varaibles in order (put 0s as well):
1
0
0
1
Enter the constants of the system:
1.1
0.0
The solutions are:
x1 = 1.1
x2 = 0.0
```

Q.7) Write a Java program to find the eigenvalues of a 2x2 matrix and verify the determinant as the product of eigenvalues

Algorithm

1) Input the elements of a 2x2 matrix.
2) Compute the trace (sum of the diagonal elements) and the determinant of the matrix.
3) Calculate the discriminant of the characteristic equation.
4) If the discriminant is negative, eigenvalues are complex. Otherwise, compute the eigenvalues using the quadratic formula.

Program

```
import java.util.Scanner;


public class EigenValues {


   public static double determinant(double[][] matrix) {
      return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
   }


   public static double[] eigenvalues(double[][] matrix) {
      double a = matrix[0][0];
      double d = matrix[1][1];


      double trace = a + d;
      double determinant = determinant(matrix);


      double discriminant = Math.pow(trace, 2) - 4 * determinant;
      if (discriminant < 0) {
         throw new ArithmeticException("The matrix has complex eigenvalues.");
      }
```

```
    double sqrtDiscriminant = Math.sqrt(discriminant);
    double lambda1 = (trace + sqrtDiscriminant) / 2;
    double lambda2 = (trace - sqrtDiscriminant) / 2;

    return new double[] {lambda1, lambda2};
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    double[][] matrix = new double[2][2];

    System.out.println("Enter the elements of the 2x2 matrix:");
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            matrix[i][j] = scanner.nextDouble();
        }
    }

    try {
        double[] eigenvalues = eigenvalues(matrix);
        double det = determinant(matrix);
        double productOfEigenvalues = eigenvalues[0] * eigenvalues[1];

        System.out.println("Eigenvalue 1: " + eigenvalues[0]);
        System.out.println("Eigenvalue 2: " + eigenvalues[1]);
        System.out.println("Determinant of the matrix: " + det);
```

```
            System.out.println("Product of the eigenvalues: " + productOfEigenvalues);


            if (Math.abs(det - productOfEigenvalues) < 1e-9) {
                System.out.println("The determinant is equal to the product of the eigenvalues.");
            } else {
                System.out.println("The determinant is NOT equal to the product of the eigenvalues.");
            }
        } catch (ArithmeticException e) {
            System.out.println(e.getMessage());
        }

        scanner.close();
    }
}
```

Sample I/O

```
C:\Users\niran\Downloads\Uni_Work\java\35_Niranjhan>java ./EigenValues.java
Enter the elements of the 2x2 matrix:
3
4
3
1
Eigenvalue 1: 5.60555127546399
Eigenvalue 2: -1.6055512754639891
Determinant of the matrix: -9.0
Product of the eigenvalues: -9.0
The determinant is equal to the product of the eigenvalues.
```